无向图中连通支配集问题的精确算法 *

周晓清 [1,2]，叶安胜 [2†]，张志强 [2]

(1. 电子科技大学 计算机科学与工程学院, 成都 611731; 2. 成都大学 信息科学与工程学院, 成都 610106)

摘　要：图 $G=(V,E)$ 的一个支配集 $D \subseteq V$ 是一个顶点子集，使得图中每一个顶点要么在 D 中，要么至少与 D 中的一个顶点相连。连通支配集问题是找到一个顶点数最小的支配集 S，并且 S 的导出子图 G[S] 是连通图。该问题是一个经典的 NP 难问题，可应用于连通设施选址、自适应网络等领域。针对无向图中连通支配集问题，仔细分析该问题的图结构性质，挖掘出若干有效的约简规则和分支规则，设计了一个分支搜索算法，并采用了测量治之方法分析算法的运行时间，最终得到了一个运行时间复杂度为 $O^*(1.93^n)$ 的精确算法。

关键词：NP 难问题；精确算法；测量治之；连通支配集问题

中图分类号：TP301.5　　doi: 10.3969/j.issn.1001-3695.2018.03.0154

Exact algorithms for connected dominating set in undirected graphs

Zhou Xiaoqing[1, 2], Ye Ansheng[2†], Zhang Zhiqiang[2]

(1. *School of Computer Science & Engineering*, *University of Electronic Science & Technology of China*, *Chengdu* 611731, *China*; 2. *College of Information Science & Engineering*, *Chengdu University*, *Chengdu* 610106, *China*)

Abstract: A dominating set $D \subseteq V$ of a graph $G=(V,E)$ is a subset of vertices, such that every vertex of G is either in D, or adjacent to at least one vertex in D. The connected dominating set problem asks to find a dominating set S with minimum number of vertices and the induced subgraph $G[S]$ of S is connected. The connected dominating set problem is a classical NP-hard problem, which could be applied to connected facility location, ad-hoc networks and many other areas. For the connected dominating set problem in undirected graphs, this paper carefully analyzes the structural properties, explores a number of effective reduction rules as well as branching rules and provides a branch-and-search algorithm. A measure-and-conquer method is also introduced to analyze the running time bound. Finally, an exact algorithm with a running time complexity of $O^*(1.93^n)$ is obtained.

Key words: NP-hard problem; exact algorithms; measure-and-conquer; connected dominating set problem

0　引言

NP 和 NP 完全的概念是 Cook[1]于 1971 年提出来的，自提出后 NP 理论得到广泛研究。当 $P \neq NP$ 时，NP 完全问题不存在多项式时间算法。对于许多 NP 难问题，Impagliazzo 等人[2]证明不存在亚指数时间算法。此外，这些问题没有或很难获得近似算法。例如，著名的 3-SAT 问题，要么为真要么为假，采用近似算法求解毫无意义。在 *P=NP* 这个目前被广泛认为不大可能的结果出现之前，要得到 NP 难问题的准确解只能采用精确算法[3]。

非平凡精确算法的历史可以追溯到 1962 年，Held 等人[4]对 TSP（traveling salesman problem）设计了运行时间为 $O(2^n n^2)$ 的动态规划算法（其中 n 为图中的顶点数），这也是该问题迄今为止最快算法之一，该问题是否存在运行时间为 $O^*(1.99^n)$ 的算法已经成为计算理论中一个著名的公开难题。另一个著名问题是最大独立集问题(maximum independent set problem)，1977 年 Tarjan 等人[5]提出了运行时间为 $O(2^{n/3})$ 的最大独立集算法。经过一系列学者的研究，1986 年 Robson[6]将该问题运行时间的上界改进到 $O(2^{0.276n})$，目前该问题的运行时间被改进到 $O^*(1.1996^n)$ [7]，然而将这个上界改进到 1.2^n 以下用了 30 多年。对于不同的 NP 难问题，目前已知的精确算法的时间复杂度之间差距较大，如前所提及到 TSP 问题和最大独立集问题。那么在时间复杂度较差的各种问题之间是否存在联系，或者当解决某个 NP 难问题是否有助于其他 NP 难问题的解决，或者是否可以以某种方式对经典 NP 完全问题进行分类后预测较差时间复杂度的算法最好能有多好，这些都是需要考虑的问题，有助

收稿日期：2018-03-08；**修回日期：**2018-04-23　**基金项目：**国家重点研发计划资助项目（2016YFB0800605）；国家自然科学基金资助项目（61370071）；四川省教育厅科研项目重点项目（15ZA0354）

作者简介：周晓清（1982-），女，乐山犍为人，博士研究生，主要研究方向为算法分析与设计、精确算法和智能计算；叶安胜（1971-），男（通信作者），教授，主要研究方向为智能计算及大数据研究（yas@cdu.edu.cn）；张志强（1976-），男，副教授，主要研究方向为智能计算及数据挖掘.

于更好的认识 NP 完全问题的计算复杂性。

对于支配的概念，1962 年 Claude[8]首次描述了现代图的支配概念。同一年，Ore[9]第一次在文献中使用了支配集（dominating set）这个术语。支配集包括顶点支配集和边支配集，与此同时，支配集问题包含顶点支配问题和边支配问题。最小顶点支配集问题（dominating set problem）是一个经典的 NP 难问题[10]，2004 年 Fomin 等人[11]设计了运行时间为 $O^*(1.9378^n)$ 的精确算法，第一次突破了 2^n 这个平凡时间界，对于该问题目前已知的最佳精确算法的运行时间是 $O^*(1.4969^n)$ [12]。对于最小边支配集问题（edge dominating set problem），Yannakakis 等人[13]证明其是 NP 难的，2008 年 Schiermeyer[14]第一次打破了 2^n 这个平凡时间界，目前该问题已知的最佳精确算法的运行时间为 $O^*(1.3160^n)$ [15]。对于本文研究的连通支配集问题（connected dominating set problem），Garey 等人[10]证明该问题是 NP 难的。2008 年 Fomin 等人[16]第一次打破了 2^n 这个平凡时间界，提出了运行时间为 $O^*(1.9407^n)$ 的精确算法。从表面来看，连通支配集问题和支配集问题紧密相连，似乎可以用解决支配集问题的方法来解决连通支配集问题。但是，实际上这样是行不通的，因为从精确算法的角度来看这两者之间具有很大的不同。事实上，连通支配集问题是属于一类称为非局部性的问题（non-local problems），通常这种非局部性问题都很难处理，例如经典的 TSP 问题。由于解决支配集问题的精确算法通常是基于问题的局部结构，不能很好的抓住全局连通的这种属性，所以很难将解决支配集的技术用来解决连通支配集。类似这种非局部性问题有反馈顶点集问题（feedback vertex set problem）、斯坦纳树问题（Steiner tree problem）等。

图 G 的一个支配集是一个顶点子集 D 使得图中的顶点如果不在 D 中那么至少与 D 中一个顶点相连。最小顶点支配集问题就是在图中找出一个顶点数最少的支配集，该问题是一个经典的 NP 难问题[10]。如果对支配集中的顶点加一些约束条件，那么最小顶点支配集问题可以转化成另外一些 NP 难问题。若要求支配集 D 中的任意两个顶点之间没有边相连，即 D 是一个独立集，那么该问题就是独立支配集问题；若要求支配集 D 中的任意两个顶点之间都有边相连，即 D 是一个团，那么该问题就是支配团问题；若要求支配集 D 的导出子图为一个连通图，该问题就是连通支配集问题。本文解决的问题就是连通支配集问题。图 G 的一个连通支配集是一个顶点子集 S，要求 S 不仅为支配集而且其导出子图是连通的。最小连通支配集问题就是在图中找出一个顶点数最小的连通支配集。该问题等价于寻找图 G 的一棵生成树，要求该生成树的叶子结点数达到最大。连通支配集在连通的设施选址、操作系统、无线自适应网络方面都有广泛的应用[17~22]。

本文针对连通支配集问题给出了运行时间为 $O^*(1.93^n)$ 的精确算法，其算法基本思想源自于 Fomin 等人[16]设计的精确算法的思想“保持连通”，即在递归过程中所产生的部分解必须是连通的，同时采用测量治之（measure and conquer）[23,24]的方法设计递归算法和分析递归算法的时间复杂度。在使用测量治之分析递归算法的时间复杂度时，文中采用了非常简单的方式来设置度量，同时对时间复杂度的分析也非常清晰，具体参见第 4 小节。文中仔细分析连通支配集问题的图结构性质，细致地挖掘了该问题相关的性质，在此基础上设计了有效的约简规则和分支规则，使得该问题得以简化。针对连通支配集问题设计的约简规则和分支规则，降低了原问题的求解规模和难度，结合启发式算法在处理连通支配集问题的具体应用中会更加高效。

1 符号和问题定义

简单图是指没有平行边和自环的图，本文涉及到的所有图均指无向简单图。设 $G=(V,E)$ 表示一个顶点集为 V 和边集为 E 的图，其中 $|V|=n$ 和 $|E|=m$。如果一个顶点子集仅包含一个元素 $\{v\}$，可以简单记为 v。对于图 G 中的任意一个顶点 v，用 $N(v)=\{u\in V: uv\in E\}$ 表示顶点 v 的开邻集，用 $N[v]=N(v)\bigcup v$ 表示顶点 v 的闭邻集。$N_2(v)$ 表示距离顶点 v 为 2 的顶点集合。$d(v)$ 表示顶点 v 的度，即图中边的一个端点为 v 的条数。在简单图中，$d(v)=|N(v)|$ 始终成立。对于一个顶点子集 X，令 $N(X)=\bigcup_{v\in X}N(v)\backslash X$。$G[X]$ 表示由顶点子集 $X\subseteq V$ 导出的子图，$G[V\setminus X]$ 也可以简写为 $G\setminus X$。如果顶点子集 X 的导出子图 $G[X]$ 是连通图，则称顶点子集 X 为连通子集。对于算法运行时间分析，用 O^* 符号省略多项式部分，如对 $f(n)$ 和 $g(n)$ 两个函数，用 $f(n)=O^*(g(n))$ 表示 $f(n)=g(n)\cdot n^{O(1)}$。

为了不失一般性，文中作以下假设：文中所提及的图均为连通图，否则该问题无解；图 G 的最小连通支配集的大小至少为 2，否则该问题可以在多项式时间内解决。

在算法的执行过程中，某些顶点一定会被包含最终解中，用 $S\subseteq V$ 表示这些顶点构成的集合，即顶点集 S 中的所有顶点都会在最终解中，其中 $|S|\geq 2$ 并且 $G[S]$ 是连通的。同时，在算法执行过程中，某些顶点一定不会在最终解中，用顶点集 $D\subseteq V$ 表示这些顶点构成的集合，即在算法执行过程中将这些顶点从图中删除。假设用 OPT 表示最优解，那么顶点集 S 中的所有顶点一定在 OPT 中，顶点集 D 中的所有顶点一定不在 OPT 中，即 $S\subseteq OPT$ 且 $D\bigcap OPT=\phi$。用 $A=V\setminus(S\bigcup D)$ 表示图 G 中即没有被选入集合 S 中也没有被选入集合 D 中的顶点，即还未被处理的顶点，称为可用顶点集。如果删除 A 中的一个顶点 v 使得问题实例无解，则称顶点 v 为必选点。如果 A 中的一个顶点 v 至少与集合 S 中的一个顶点相邻，那么称顶点 v 为候选点，否则称为自由点。用 C 表示候选点构成的集合，$C=\bigcup_{v\in S}N(v)\bigcap A$。用 F 表示自由点构成的集合，$F=A\setminus C$。如果图 G 中的一个顶点 v 至少与集合 S 中的一个顶点相邻，那么称集合 S 支配顶点 v。本文考虑的是如下一个带约束的最小连通支配集问题：

带约束的最小连通支配集问题

输入:一个无向图 $G=(V,E)$ 和 G 的两个顶点子集 $S,D\subseteq V$，其中导

出子图 $G[S]$ 为连通图。

问题: 找出图 G 的一个最小顶点子集 $Y \subseteq V$ 满足如下性质：Y 是一个支配集，$S \subseteq Y$ 且 $D \cap Y = \phi$，导出子图 $G[Y]$ 为连通图。

可以看出，当 S 包含图中任意两个相邻点 u 和 v 时，带约束的最小连通支配集问题就是求包含 u 和 v 的最小连通支配集。本文研究这种带约束条件的问题，其主要原因是在分支搜索算法中某些分支会要求将一些顶点保留在最后的解中，那么可以直接将这些顶点加入 S 中。对于带约束的最小连通支配集问题，任意一个包含 S 中所有顶点且是连通支配集的顶点子集称为该问题的一个可行解。当一个可行解的大小达到最小时，就称该可行解为最优解，在本文中将一个最优解简称为一个解。

2 算法设计

2.1 初步思想

由于最小连通支配集问题具有连通的全局属性，算法设计的初步思想是猜测图中的任意两个相邻顶点 v_1 和 v_2 可能在某些最优解中，然后任意拿出 A 中一个候选点 v 分两种情况进行讨论：要么 v 在 Y 中；要么 v 不在 Y 中。如果 v 在 Y 中，那么直接将 v 加入 S 中；如果 v 不在 Y 中，那么从图中直接删除 v 。如此不断的进行分支搜索，直到 A 为空集或 A 中不存在任何候选点，这样就可以解决该问题。但是这种算法思想在理论上分析只能得到 $O^*(2^n)$ 的运行时间界，很长时间都未有人突破这个简单的运行时间界，直到 Fomin 等人用测量治之方法来解决该问题[16]。在解决连通支配集问题时，Fomin 等人的主要思想还是不断选出一个顶点 v 分两种情况进行讨论：要么将 v 保留到最后的解中，要么从图中删除，只是将一个顶点加入解时一定要对解“保持连通”。本文同样遵循这种“保持连通”的思想，并且在处理 A 中的候选点进行了更为细致的思考。在考虑将 A 中的一个候选顶点 v 加入解的目的有两个：一是为了支配 v 的所有自由点邻居；二是为了连通 v 的某一个自由点邻居。对于前一种情况，当将 v 加入解后，可以简单的将 v 的所有自由点邻居直接删除；对于后一种情况，当将 v 加入解后，可以将 v 的某一个自由点邻居加入解中。本文就是基于这样一个思想，不断的处理 A 中的候选点，当 $A=\phi$ 或者不存在候选点时，算法要么找到一个解，要么解不存在，最后采用测量治之方法对算法的进行分析和设计，从而将算法的运行时间界进一步进行改进。

2.2 定理和性质

在多项式时间内如果通过实例 I 的一个解 Y 可以构造出另一个问题实例 I' 的一个解 Y'，那么称问题实例 I 等价问题实例 I'，反之亦然。下面将先证明问题实例的等价性质，在等价性质的基础上定义一些约简操作，通过这些约简操作简化原问题实例。

定理 1 对于一个带约束的最小连通支配集问题实例 $I=(G,S,D)$，若 I 存在一个解不包含某个顶点 v，则删除 v 以后的实例 $I'=(G\setminus v,S,D\cup\{v\})$ 的任意一个解是原问题实例 I 的一个解；若 I 存在一个解包含某个顶点 u，则将 u 加入 S 以后的实例 $I''=(G,S\cup\{u\},D)$ 的任意一个解是原问题实例 I 的一个解。

证明 设 $G'=G\setminus v$，Y' 为 I' 的任意一个最优解。因为 G' 是 G 的一个子图，所以 Y' 也是 I 的一个可行解。假设实例 I 的一个最优解 Y 不包含顶点 v，那么 Y 仍然是 I' 的一个可行解。这种情况下有

$$|Y|\le|Y'|.$$

此时 Y' 是 I 的一个最优解。

令 Y'' 为 I'' 的任意一个最优解。因为 Y'' 的导出子图为连通图，因此 Y'' 是 I 的一个可行解。假设实例 I 的一个最优解 Y 包含顶点 u，那么 Y 仍然是 I'' 的一个可行解。这种情况下有

$$|Y|\le|Y''|.$$

此时 Y'' 是 I 的一个最优解。

通过定理 1 可以得到，如果知道某个顶点在一个最优解中时，根据该定理可以将顶点加入 S 中，如果知道某个顶点不在一个最优解时，那么可以直接删除该顶点。

性质 1 对于一个带约束的最小连通支配集问题实例 $I=(G,S,D)$，如果存在一个候选顶点 v 是必选点，则将顶点 v 加入 S 。

由于顶点 v 为必选点，删除 v 后问题实例 I 无解，那么 v 必须在问题实例最后的解中，根据定理 1 显然可知性质 1 的正确性。

性质 2 如果存在两个候选顶点 v 和 u，根据性质 1 可知这两个顶点都不是必选点，假设 $N(v)\cap F \subseteq N(u)\cap F$，那么将顶点 v 加入 D 中。

假设存在一个最优解 OPT 包含顶点 v，由于顶点 v 所支配的自由点全部被顶点 u 所支配，而顶点 v 和 u 本身已经被支配，那么存在另一个不包含 v 且大小至多为 $|OPT|$ 的可行解 $OPT'=OPT\setminus\{v\}\cup\{u\}$，所以顶点 v 可以删除。

性质 3 如果存在一个可用点 v 不支配任何的自由点，那么将顶点 v 加入 D 中。

由于顶点 v 不支配任何的自由点，所以 v 的所有邻居都为候选点，且所有候选点都与 S 连通，因此从任何可行解中移除 v 都能保持解的可行性。

根据性质 3 对图进行约简后，可以很容易得到，如果图中存在一个候选点 v，那么 v 至少与一个自由点相邻，否则可直接将 v 加入 D 中。

性质 4 假设图 G 中存在一个候选点 v 支配一个可用点 w，如果将 v 加入 S 后，可用点 w 不再支配任何的自由点，那么可以将 w 加入 D 中。

因为将候选点 v 加入 S 后，可用点 w 不再支配任何的自由点，根据性质 3 可知，当一个可用点不再支配任何的自由点时，可将该顶点加入 D 中。

根据性质 4 对图进行约简后，可以很容易的得到，如果图 G 中存在一个候选点 v 支配了一个可用点 w，那么 w 必然支

配另外的自由点 $u \notin N(v)$。

性质 5　假设图 G 中存在一个候选点 v 仅支配一个自由点 w，设 $U=\{u_1,u_2,...,u_k\}=N(w)\cap A\setminus N[v]$，如果实例 $I=(G,S,D)$ 存在一个最优解 OPT 包含 v 但不包含 w，那么可以将 U 加入 D 中。

证明　设 OPT 为实例 $I=(G,S\cup\{v\},D\cup\{w\})$ 的最优解，那么 $v\in OPT$ 和 $w\notin OPT$。为了构造矛盾假设 $u_i\in U$ 且 $u_i\in OPT$，那么 $OPT=O'\cup\{v,u_i\}$（O' 为某些合适的顶点集合）。由于 $w\notin OPT$，$OPT'=O'\cup u_i$ 仍然连通，而 v 只支配 w，w 又被 u_i 支配，所以 OPT' 是一个支配集，因此 OPT' 是一个大小为 $|OPT|-1$ 的连通支配集，这与 OPT 是最优解相矛盾，见图 1 中的情况 1。

性质 6　假设图 G 中存在一个候选点 v 仅支配一个自由点 w，且 w 也仅支配一个自由点 $u\notin N(v)$，如果实例 $I=(G,S,D)$ 存在一个包含 v 但不包含 w 的最优解 OPT，那么存在另一个不包含 v 的最优解 OPT'。

证明　设 OPT 为实例 $I=(G,S\cup\{v\},D\cup\{w\})$ 的最优解，那么 $v\in OPT$ 和 $w\notin OPT$。假设 $u\in OPT$，那么存在一个可行解 $O=OPT\setminus\{v\}$，这与 OPT 是最优解相矛盾，所以 $u\notin OPT$。由于最优解 OPT 会支配 u，所以 OPT 中至少包含一个 u 的除 w 之外的邻居 $x\subseteq N(u)\setminus\{w\}$，那么 $OPT=O'\cup\{v,x\}$（O' 为某些合适的顶点集合）。由于 v 是候选点（v 已经被支配），且 v 只支配顶点 w，而 w 也只支配 u，所以可以将最优解 OPT 中的 v 换成 u，得到另一个可行解 $OPT'=O'\cup\{u,x\}$，又 $|OPT'|=|OPT|$，那么 OPT' 是最优解，所以可以抛弃 v，见图 1 中的情况 2。

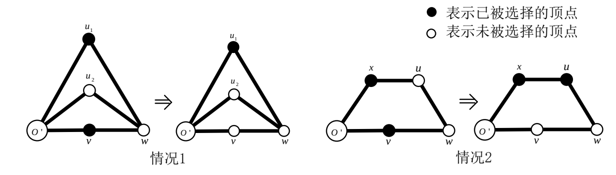


图 1　图 G 中候选点 v 仅支配一个自由点 w 的情况

定理 2. 给出一个实例 $I=(G,S,D)$，假设图 G 存在一个候选点 v 仅支配一个自由点 w，实例 I 存在一个最优解 Y 使得如下成立：

a）如果 w 仅支配一个自由点 $u\notin N(v)$，要么 $v\notin Y$，要么 $\{v,w\}\in Y$；

b）如果 w 至少支配两个自由点，设自由点集合为 $U=\{u_1,u_2,...,u_k\}=N(w)\cap A\setminus N[v]$，其中 $k\geq 2$，那么要么 $v\notin Y$，要么 $\{v,w\}\in Y$，要么 $v\in Y$ 且 $\{w\cup U\}\notin Y$。

证明：a）设 Y' 为一个问题实例的最优解，假设 Y' 包含顶点 v，否则 a）直接成立。Y' 必然不包含顶点 u，否则存在一个包含 u 但不包含 v 可行解 Y''，使得 $|Y''|<|Y'|$，这与 Y' 是最优解相矛盾。又 Y' 为最优解必然支配顶点 u，所以至少 u 的一个邻居在 Y'，如果 w 在 Y' 中，a) 直接成立，如果 w 不在 Y' 中，根据性质 6 可知，可以找到另外一个不包含 v 的最优解，所以 a）也成立。

b）设 Y' 为一个问题实例的最优解，假设 Y' 包含顶点 v，否则 b）直接成立。Y' 必然不包含 u_i，假设 Y' 包含 u_i，那么存在一个包含 u_i 但不包含 v 可行解 Y''，使得 $|Y''|<|Y'|$，这与 Y' 是最优解相矛盾，所以 Y' 必然不包含 u_i。又 Y' 为最优解必然支配顶点 u_i，所以 u_i 的邻居中至少有一个在 Y'，如果 w 在 Y' 中，b）直接成立，如果如果 w 不在 Y' 中，根据性质 5 可知，可以直接将 U 加入 D 中，所以 b）成立。

综上所述，定理 2 成立。

根据定理 2 对图进行约简后，可以很容易得到，如果图中存在一个候选点 v，那么 v 至少与两个自由点相邻。

性质 7　假设图 G 存在一个候选点 v 仅支配了两个自由点 $\{w_1,w_2\}$，设 $U_i=\{u_{i,1},u_{i,2},...,u_{i,k}\}=N(w_i)\cap A\setminus N[v]$，如果实例 $I=(G,S,D)$ 存在一个最优解 OPT 包含 v 但不包含 $\{w_1,w_2\}$，那么或者将 U_1 加入 D 中，或者将 U_2 加入 D 中。

证明　设 OPT 为实例 $I=(G,S\cup\{v\},D\cup\{w_1,w_2\})$ 的最优解，那么 $v\in OPT$ 和 $\{w_1,w_2\}\notin OPT$。为了构造矛盾假设 $\{u_{1,j},u_{2,l}\}\in OPT$，其中 $j,l=\{1,2,...,k\}$，那么 $OPT=O'\cup\{v,u_{1,j},u_{2,l}\}$（$O'$\$为某些合适的顶点集合）。由于顶点 $\{w_1,w_2\}\notin OPT$，$OPT'=O'\cup\{u_{1,j},u_{2,l}\}$ 仍然连通，而 v 只支配 $\{w_1,w_2\}$，$\{w_1,w_2\}$ 又分别被 $\{u_{1,j},u_{2,l}\}$ 支配，所以 OPT' 是一个支配集，因此 OPT' 是一个大小为 $|OPT|-1$ 的连通支配集，这与 OPT 是最优解相矛盾，见图 2 中的情况 3。

性质 8　假设图 G 存在一个候选点 v 仅支配了两个自由点 $\{w_1,w_2\}$，且 w_1 和 w_2 仅支配一个自由点，分别为 u_1 和 u_2，如果实例 $I=(G,S,D)$ 存在一个包含 v 但不包含 w_1 和 w_2 的最优解 OPT，那么要么该最优解 OPT 不包含 u_1 和 u_2，要么存在另一个不包含 v 最优解 OPT'。

证明　设 OPT 为实例 $I=(G,S\cup\{v\},D\cup\{w_1,w_2\})$ 的最优解，那么 $v\in OPT$ 和 $\{w_1,w_2\}\notin OPT$。反证 OPT 不包含 u_1 和 u_2，假设 $\{u_1,u_2\}\in OPT$，那么存在另外一个可行解 $O=OPT\setminus\{v\}$，这与 OPT 是最优解相矛盾，所以 $\{u_1,u_2\}\notin OPT$。假设 u_1 和 u_2 中有一个是属于 OPT，假定 $u_1\in OPT$ 且 $u_2\notin OPT$，由于最优解 OPT 会支配 u_2，所以 OPT 中至少包含一个 u_2 的除 w_2 之外的邻居 $x\in N(u_2)\setminus\{w_2\}$，那么 $OPT=O'\cup\{v,x,u_1\}$（O' 为某些合适的顶点集合）。由于 v 是候选点（v 已经被支配），w_1 被 $u_1\in OPT$ 支配，那么将最优解 OPT 中 v 换成 u_2，得到另一个可行解 $OPT'=O'\cup\{u_2,x,u_1\}$，又 $|OPT'|=|OPT|$，那么 OPT' 是一个不包含 v 的最优解，同理可以证 $u_2\in OPT$ 且 $u_1\notin OPT$ 的情况，所以可以抛弃 v，见图 2 中的情况 4。

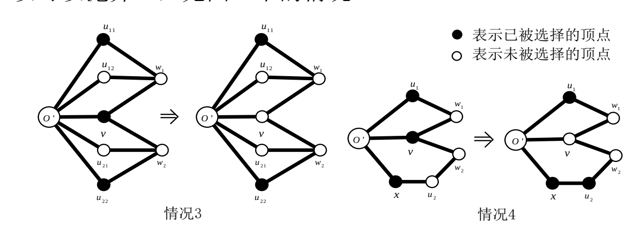


图 2　图 G 中候选点 v 仅支配两个自由点 w_1 和 w_2 的情况

定理 3　给出一个实例 $I=(G,S,D)$，假设图 G 存在一个

候选点 v 仅支配两个自由点 $\{w_1,w_2\}$，实例 $I=(G,S,D)$ 存在一个最优解 Y 使得如下成立：

a）如果 w_i 仅支配一个自由点 $u_i \notin N(v)$，即 w_1 支配 u_1 和 w_2 支配 u_2，那么要么 $v \notin Y$，要么 $\{v,w_1\} \in Y$，要么 $\{v,w_2\} \in Y$ 且 $w_1 \notin Y$，要么 $v \in Y$ 且 $\{\{w_1,w_2\} \cup u_1 \cup u_2\} \notin Y$。

b）如果 w_i 至少支配两个自由点，设 w_i 支配的自由点集合为 $U_i=\{u_{i,1},u_{i,2},\dots,u_{i,k}\}=N(w_i)\cap A \setminus N[v]$，其中 $k \ge 2$，那么要么 $v \notin Y$，要么 $\{v,w_1\} \in Y$，要么 $\{v,w_2\} \in Y$ 且 $w_1 \notin Y$，要么 $v \in Y$ 且 $\{\{w_1,w_2\} \cup U_1\} \notin Y$，要么 $v \in Y$ 且 $\{\{w_1,w_2\} \cup U_2\} \notin Y$。

证明　a）设 Y' 为一个问题实例的最优解，假设 Y' 包含顶点 v，否则 a）直接成立。假设 w_1 和 w_2 相邻，根据性质 4 可知，这种情况不会发生。那么，考虑 w_1 和 w_2 不相邻情况。假设 w_1 和 w_2 共同支配了一个自由点 u，那么在 Y' 中将 v 替换成 u，得到一个不包含 v 的可行解 Y''，又 $|Y''| \le |Y'|$，所以 Y'' 是最优解。由于在抛弃 v 的这个分支 $(G\setminus\{v\},S,D\cup\{v\})$ 里已经考虑了该情况，所以 w_1 和 w_2 共同支配一个自由点的这种情况可以不用考虑。那么考虑 w_1 和 w_2 分别支配不同自由点 u_1 和 u_2 时的情况，如果 $\{w_1,w_2\}$ 都不在 Y' 中，根据性质 8 可知，要么最优解同时不包含 u_1 和 u_2，要么存在另一个不包含 v 的最优解，从而 a）成立。如果 w_1 和 w_2 都在 Y' 中，存在一个可行解 $Y'''=Y'\setminus\{v\}$，这与 Y' 是最优解相矛盾，所以 w_1 和 w_2 不可能同时在 Y' 中。如果 w_1 在 Y' 中，a）直接成立。如果 w_2 在 Y' 中，必然抛弃 w_1，a）也成立。

b）设 Y' 为一个问题实例的最优解，假设 Y' 包含顶点 v，否则 b）直接成立。假设 w_1 和 w_2 相邻，那么在 Y' 中将 v 替换成 w_1，得到一个不包含 v 的可行解 Y''，又 $|Y''| \le |Y'|$，所以 Y'' 是最优解。同样由于在抛弃 v 的这个分支 $(G\setminus\{v\},S,D\cup\{v\})$ 里已经考虑了该情况，所以 w_1 和 w_2 相邻这种情况可以不用考虑。那么，考虑 w_1 和 w_2 不相邻情况。假设 w_1 和 w_2 共同支配了一个自由点 u，那么在 Y' 中将 v 替换成 u，得到一个不包含 v 的可行解 Y''。同样，这种情况不用考虑。那么考虑 w_1 和 w_2 支配的自由点各不相同并且至少支配两个自由点时情况，设 w_i 支配的自由点集合为 $U_i=\{u_{i,1},u_{i,2},\dots,u_{i,k}\}=N(w_i)\cap A\setminus N[v]$。如果 $\{w_1,w_2\}$ 都不在 Y' 中，根据性质 7 可知，或者将 U_1 加入 D 中，或者将 U_2 加入 D 中，从而 b）成立。如果 w_1 和 w_2 都在 Y' 中，存在一个可行解 $Y'''=Y'\setminus\{v\}$，这与 Y' 是最优解相矛盾，所以 w_1 和 w_2 不可能同时在 Y' 中。如果 w_1 在 Y' 中，b）直接成立。如果 w_2 在 Y' 中，必然抛弃 w_1，b）也成立。

综上所述，定理 3 成立。

根据定理 3 对图进行约简后，可以很容易得到，如果图中存在一个候选点 v，那么 v 至少与三个自由点相邻。

2.3　算法设计

基于以上的定理和性质，下面进行算法的详细设计，其主要步骤参见算法 1。

算法 1.　算法 $MCD(G,S,D)$

输入：一个无向图 $G=(V,E)$ 和 G 的两个顶点子集 $S,D \subseteq V$，其中 $S=\{v_1,v_2 \mid v_1v_2 \in E\}$ 且 $G[S]$ 为连通图。

输出：图 G 中满足 $S \subseteq Y$ 的最小连通支配集 Y。

1. **如果** $G[S]$ 不为连通图，则停止算法，返回 ϕ。
2. **否则如果** $A=\phi$ 且 $G[S]$ 为连通图，则返回 S 作为解。
3. **否则如果** $A \ne \phi$，且不存在任何的候选点，则返回 ϕ。
4. **否则如果**图中存在一个候选点 v 是必选点，则返回 $MCD(G,S\cup\{v\},D)$。
5. **否则如果**图中存在两个候选点 v 和 w（v 和 w 都不是必选点），且 $N(v)\cap F \subseteq N(w)\cap F$，则返回 $MCD(G\setminus v,S,D\cup\{v\})$。
6. **否则如果**图中存在一个可用点 v 不支配任何的自由点，即顶点 v 的所有邻居全部都被支配，则返回 $MCD(G\setminus v,S,D\cup\{v\})$。
7. **否则如果**图中存在一个候选点 v 支配一个可用点 w，选择顶点 v 后，w 不支配任何自由点，则返回 $MCD(G\setminus v,S,D\cup\{v\})$ 和 $MCD(G\setminus w,S\cup\{v\},D\cup\{w\})$ 中的较大解。
8. **否则如果**当前图中存在一个候选点 v，v 仅支配一个自由点 w，而 w 也仅支配一个自由点 $u \notin N(v)$，则返回 $MCD(G\setminus v,S,D\cup\{v\})$ 和 $MCD(G,S\cup\{v,w\},D)$ 中较大的解。
9. **否则如果**当前图中存在一个候选点 v，v 仅支配一个自由点 w，w 至少支配两个自由点，设 w 支配的自由点集合为 $U=\{u_1,u_2,\dots,u_k\}=N(w)\cap A\setminus N[v]$，其中 $k \ge 2$，则返回 $MCD(G\setminus v,S,D\cup\{v\})$，$MCD(G,S\cup\{v,w\},D)$ 和 $MCD(G\setminus\{w,U\},S\cup\{v\},D\cup\{w,U\})$ 中较大的解。
10. **否则如果**当前图中存在一个候选点 v，v 仅支配两个自由点 w_1 和 w_2，w_1 和 w_2 仅支配一个自由点分别为 u_1 和 u_2，其中 $\{u_1,u_2\} \notin N(v)$，则返回 $MCD(G\setminus v,S,D\cup\{v\})$，$MCD(G,S\cup\{v,w_1\},D)$，$MCD(G\setminus\{w_1\},S\cup\{v,w_2\},D\cup\{w_1\})$，$MCD(G\setminus\{w_1,w_2,u_1,u_2\},S\cup\{v\},D\cup\{w_1,w_2,u_1,u_2\})$ 中较大的解。
11. **否则如果**当前图中存在一个候选点 v，v 支配两个自由点 w_1 和 w_2，w_1 和 w_2 至少支配了两个自由点，设 w_i 支配的自由点集合为 $U_i=\{u_{i,1},u_{i,2},\dots,u_{i,k}\}=N(w_i)\cap A\setminus N[v]$，其中 $k \ge 2$，则返回 $MCD(G\setminus v,S,D\cup\{v\})$，$MCD(G,S\cup\{v,w_1\},D)$，$MCD(G\setminus\{w_1\},S\cup\{v,w_2\},D\cup\{w_1\})$，$MCD(G\setminus\{w_1,w_2,U_1\},S\cup\{v\},D\cup\{w_1,w_2\}\cup U_1)$ 和 $MCD(G\setminus\{w_1,w_2,U_2\},S\cup\{v\},D\cup\{w_1,w_2\}\cup U_2)$ 中较大的解。
12. **否则**当前图中存在一个候选点 v，且 v 至少有三个自由点邻居时，则返回 $MCD(G\setminus v,S,D\cup\{v\})$ 和 $MCD(G,S\cup\{v\},D)$ 中较大的解。

文中用 $MCD(G,S,D)$ 表示算法，输入部分为一个带约束的最小连通支配集问题的实例，包括一个无向连通图 $G=(V,E)$ 和 G 的两个顶点子集 $S,D \subseteq V$，其中导出子图 $G[S]$ 为连通图。算法开始时，设 S 中包含图中任意一条边的两个顶点 v_1 和 v_2，D 为空集，即 $MCD(G,\{v_1,v_2\},\phi)$。算法输出的是一个包含 S 中所有顶点的最小连通支配集 Y。为了求出最小连通支配集，该算法需要执行 $O(n^2)$ 次。

该算法主要包含 12 个步骤，用递归算法的形式进行描述，即每一个步骤都调用算法本身。第 1 步当 $G[S]$ 不为连通图时，则停止算法，返回 ϕ，这一步显然正确。第 2 步当 $A=\phi$ 且 $G[S]$ 为连通图时，返回 S 作为解，这一步显然正确。第 3 步当 A 不为空集但不存在任何的候选点，返回空集作为问题解，这一步也显然正确。第 4 步主要是将必选点加入 S 中，这一步的正确性由性质 1 给出。第 5 步当存在两个候选点 v 和 u，如果 v 所支配自由点全部被 u 所支配那么删除 v，这一步的正确性由性质 2 给出。第 6 步从图中删除一些不支配任何自由点的可用点，该步的正确性由性质 3 给出。第 7 步则是考虑存在一个候选点 v 支配另一个可用点 w，当选择 v 后，w 不支配任何自由点的情况。算法分为两个分支，第一个分支将 v 从图中删除，第二分支则是将 v 加入 S 中并且将 w 删除。其正确性由性质 4 给出。第 8 步则是考虑存在一个候选点 v 仅支配一个自由点 w，w 也仅支配一个自由点 $u\notin N(v)$ 时的情况。算法分为两个分支，第一个分支将 v 从图中删除，第二个分支将 v 和 w 加入 S 中，其正确性由定理 2 给出。在第 9 步则是考虑存在一个候选点 v 仅支配一个自由点 w，w 至少支配两个自由点时的情况，设 w 支配的自由点集合为 $U=\{u_1,u_2,\ldots,u_k\}=N(w)\cap A\setminus N[v]$，其中 $k\ge 2$，算法分为三个分支，第一个分支将 v 从图中删除，第二个分支将 v 和 w 加入 S 中，第三个分支将 v 加入 S 中同时删除 $\{w,U\}$，其正确性由定理 2 给出。第 10 步和第 11 步则是考虑存在一个候选点 v 支配两个自由点 w_1 和 w_2 的情况。第 10 步中，w_1 和 w_2 分别仅支配一个自由点 u_1 和 u_2，算法分为四个分支，第一个分支将 v 从图中删除，第二个分支将 v 和 w_1 加入 S 中，第三个分支将 v 和 w_2 加入 S 中同时删除 w_1，第四个分支将将 v 加入 S 中同时删除 $\{w_1,w_2,u_1,u_2\}$，其正确性由定理 3 给出。第 11 步中，w_1 和 w_2 至少支配两个自由点，设 w_i 支配的自由点集合为 $U_i=\{u_{i,1},u_{i,2},\ldots,u_{i,k}\}=N(w_i)\cap A\setminus N[v]$，其中 $k\ge 2$，算法分为五个分支，第一个分支将 v 从图中删除，第二个分支将 v 和 w_1 加入 S 中，第三个分支将 v 和 w_2 加入 S 中同时删除 w_1，第四个分支将 v 加入 S 中同时删除 $\{w_1,w_2,U_1\}$，第五个分支将 v 加入 S 中同时删除 $\{w_1,w_2,U_2\}$，其正确性由定理 3 给出。第 12 步则是做简单的分支，要么删除这个候选点，要么将其放入 S 中，这一步显然正确。可以看出，只要在图中 $A=V\setminus(S\cup D)$ 不为空集，3 至 12 步中至少有一步可以执行，而当 $A=\phi$ 且 $G[S]$ 为连通时，执行第 2 步找到解集，当 $G[S]$ 不连通时，执行 1 步停止算法。由以上的分析可以得到该算法的正确性，以下则是算法运行时间的分析。

3 基于测量治之的算法运行时间分析

以上是采用分支搜索算法设计的算法，这种算法包含约简和分支这两种操作。根据以上的算法可知，算法的第 1,2,3,4,5 和 6 步只进行了约简操作，第 7,8,9,10,11 和 12 步进行了分支操作。由于约简操作不会指数级的增加算法的运行时间，所以下面仅对算法的分支操作进行分析。

采用测量治之方法对算法的运行时间复杂度进行分析时，首先需要确定衡量问题大小的度量，然后根据设定的度量为算法的所有分支操作设计递归关系式。对于带约束的最小连通支配集问题的一个实例 (G,S,D)，定义衡量问题大小的度量为：

$$\mu(G,S,D)=\alpha\,|N(S)|+|V\setminus(S\cup D\cup N(S))|,$$

其中 $0<\alpha<1$ 是一个实数，α 的值最后会根据所有分支操作列出的递归关系式计算出来，其具体值会在后面给出。根据度量的设定可知，集合 S 和 D 中的顶点权重设为 0，候选集 C 中的顶点权重设为 α，其余的顶点权重设为 1，然后将图中所有顶点的权重之和 $\mu(G,S,D)$ 作为度量。显然，在算法执行的过程中，度量 $\mu(G,S,D)$ 的值在任何时候都不会大于图中的顶点数。

下面分析在设定的度量下各分支操作产生的递归关系式，文中用 $C(\mu)$ 表示算法在度量为 μ 的问题实例上产生的搜索树大小的上界。

第 7 步　在这一步中第一个分支删除候选点 v，度量减少 α。第二个分支将 v 加入 S 中并且将 w 删除，度量减少 $\alpha+1$，又 w 至少支配一个自由点，且该自由点同时也被 v 支配，当将 v 加入 S 后，其所支配的自由点变成候选点，度量至少减少 $1-\alpha$，所以第二分支至少减少 2。这一步可以得到如下递归关系式

$$C(\mu)\le C(\mu-\alpha)+C(\mu-2).$$

第 8 步　在这一步中第一个分支删除候选点 v，度量减少 α。第二个分支将 v 加入 S 中并且也将 w 加入解中，度量减少 $1+\alpha$，w 的邻居 u 变成候选点，度量减少 $1-\alpha$，这样第二个分支减少 $1+\alpha+1-\alpha=2$。这一步可以得到如下递归关系式

$$C(\mu)\le C(\mu-\alpha)+C(\mu-2).$$

第 9 步　在这一步中第一个分支删除候选点 v，度量减少 α。第二个分支将 v 加入 S 中，并且将 w 也加入 S 中，由于 w 至少支配两个自由点，当把 w 加入 S 后，w 所支配的自由点变成了候选点，这样第二个分支的度量至少减少 $\alpha+1+2(1-\alpha)=3-\alpha$。第三个分支，将 v 加入 S 的目的是为了支配 w，所以将 v 加入 S 后，将 w 删除，同时删除 w 的自由点邻居，这样度量至少减少 $3+\alpha$。这一步可以得到如下递归关系式

$$C(\mu)\le C(\mu-\alpha)+C(\mu-(3-\alpha))+C(\mu-(3+\alpha)).$$

第 10 步　在这一步中第一个分支删除候选点 v，度量减少 α。第二个分支将 v 加入 S，其目的是为了连通 w_1，所以将 v 和 w_1 加入 S，度量至少减少 $3-\alpha$。第三个分支将 v 加入 S，其目的是为了连通 w_2，所以将 v 和 w_2 加入 S，同时删除 w_1，度量至少减少 3。第四个分支将 v 加入 S，其目的是为了支配 w_1 和 w_2，所以将 v 加入 S，同时删除 $\{w_1,w_2,u_1,u_2\}$，度量至少减少 $4+\alpha$。这一步可以得到如下递归关系式

$$C(\mu)\le C(\mu-\alpha)+C(\mu-(3-\alpha))$$
$$+C(\mu-3)+C(\mu-(4+\alpha)).$$

第 11 步　在在这一步中第一个分支删除候选点 v，度量

减少 α 。第二个分支将 v 加入 S，其目的是为了连通 w_1，所以将 v 和 w_1 加入 S，度量至少减少 $4-2\alpha$ 。第三个分支将 v 加入 S，其目的是为了连通 w_2，所以将 v 和 w_2 加入 S，同时删除 w_1，度量至少减少 $4-\alpha$ 。第四个分支将 v 加入 S，其目的是为了支配 w_1 和 w_2，所以将 v 加入 S，同时删除 w_1、w_2 以及 U_1，度量至少减少 $4+\alpha$ 。第五个分支将 v 加入 S，其目的是为了支配 w_1 和 w_2，所以将 v 加入 S，同时删除 w_1, w_2 以及 U_2，度量至少减少 $4+\alpha$ 。这一步可以得到如下递归关系式

$$C(\mu)\le C(\mu-\alpha)+C(\mu-(4-2\alpha))+C(\mu-(4-\alpha))$$
$$+C(\mu-(4+\alpha))+C(\mu-(4+\alpha)).$$

第 12 步 这一步对 v 进行简单分支，要么将 v 加入 S 中要么删除。由于此时至少支配 3 个自由点，当 v 加入 S 后，至少 3 个自由点变成候选点，其度量至少减少 $\alpha+3(1-\alpha)=3-2\alpha$ 。这一步可以得到如下递归关系式

$$C(\mu)\le C(\mu-\alpha)+C(\mu-(3-2\alpha)).$$

将上述所有递归关系式作为约束条件组成一个拟凸规划的约束条件，目标是使以上递归关系式中最大分支因子最小，其中 α 为变量。利用[25]中介绍的方法对这个拟凸规划求解得到，当 $\alpha=0.8644$ 时，上述递归关系的最大分子因子为 1.93，因此该算法运行时间上界为 $O^*(1.93^{\mu})$ 。由于在初始图上 $\mu\le n$，因此带约束的最小连通支配集问题可以在 $O^*(1.93^n)$ 时间内被解决，得到如下定理。

定理 4 最小连通支配集问题可以在 $O^*(1.93^n)$ 时间内解决。

4 结束语

针对无向图中最小连通支配集问题，本文采用分支搜索方法和测量治之方法分析和设计了一个运行时间为 $O^*(1.93^n)$ 的精确算法。文中首先分析该问题的一些特性，设计了一些约简规则简化问题实例，并在此基础上证明了一些重要的定理，最终设计出一个基于分支搜索的递归算法。文中算法的基本思想是保持解的连通性，其核心思想是在处理候选点时，对其加入解的作用进行区分对待，最后利用测量治之方法对算法的运行时间进行分析。对于连通支配集问题在一些实际问题中的应用时，文中的一些性质和定理可以在一定程度上降低原问题的求解规模和难度，因此可以将其同启发式算法相结合对该问题进行求解，达到加快求解该问题的速度和提高求解效果的目的。文中使用的测量治之方法是一种非常重要的算法分析方法，它改进了许多重要 NP 难问题的精确算法的运行时间界，对于更多的 NP 难问题该方法在其精确算法中的应用正在进一步拓展中。

参考文献：

[1] Cook S A. The complexity of theorem-proving procedures [C]// Proc of the 3rd ACM Symposium on the Theory of Computing. New York: ACM Press, 1971: 151-158.

[2] Impagliazzo R, Paturi R, Zane F. Which problems have strongly exponential complexity? [J]. Journal of Computer and System Sciences, 2001, 63 (4): 512–530, 2001.

[3] 路纲，周明天，唐勇，等. 任意图支配集精确算法回顾 [J]. 计算机学报, 2010, 33 (6): 1073-1087. (Lu Gang, Zhou Mingtian, Tang Yong, *et al.* A survey on exact algorithms for donminating set related problems in arbitrary graphs [J]. Chinese Journal of Computers, 2010, 33 (6): 1073-1087.

[4] Held M, Karp R M. A dynamic programming approach to sequencing problems [J]. Journal of SIAM, 1962, 10 (1): 196-210.

[5] Tarjan R, Trojanowski A. Finding a maximum independent set [J]. SIAM Journal on Computing, 1977, 6 (3): 537-546.

[6] Robson J M. Algorithms for maximum independent sets [J]. Journal of Algorithms, 1986, 7 (3): 425-440.

[7] Mingyu Xiao, Nagamochi H. Exact algorithms for maximum independent set [C]// Proc of the 24th International Symposium on Algorithms and Computation. Berlin: Springer Publishing Company, 2013: 328-338.

[8] Claude B. The theory of graphs and its applications [J]. Bulletin of Mathematical Biophysics, 1962, 24 (4): 441-443.

[9] Ore O. Theory of Graphs [M]. Providence: American Mathematical Society, 1962.

[10] Garey M R, Johnson D S. Computers and intractability: a guide to the theory of NP-completeness [M]. New York: W. H. Freeman, 1979.

[11] Fomin F V, Kratsch D, Woeginger G J. Exact (exponential) algorithms for the dominating set problem [C]// Proc of the 30th Workshop on Graph Theoretic Concepts in Computer Science. Berlin: Springer, 2004: 245-256.

[12] Rooij J M M V, Bodlaender H L. Exact algorithms for dominating set [J]. Discrete Applied Mathematics, 2011, 159 (17): 2147-2164.

[13] Yannakakis M, Gavril F. Edge dominating sets in graphs [J]. SIAM Journal on Applied Mathematics, 1980, 38 (3): 364-372.

[14] Schiermeyer I. Efficiency in exponential time for domination-type problems [J]. Discrete Applied Mathematics, 2008, 156 (17): 3291-3297.

[15] Xiao M, Nagamochi H. A refined exact algorithm for edge dominating set [J]. Theoretical Computer Science, 2014, 560: 207–216.

[16] Fomin F V, Grandoni F, Kratsch D. Solving connected dominating set faster than [J]. Algorithmica, 2008, 52 (2): 153-166.

[17] Ugurlu O, Tanir D, Nuri E. A better heuristic for the minimum connected dominating set in ad hoc networks [C]// Proc of IEEE International Conference on Application of Information and Communication Technologies. New York: IEEE, 2017: 1-4.

[18] Baïou M, Barahona F. The dominating set polytope via facility location [M]. Berlin: Springer International Publishing, 2014: 38-49.

[19] Mohanty J P, Mandal C, Reade C, *et al.* Construction of minimum connected dominating set in wireless sensor networks using pseudo dominating set [J]. Ad Hoc Networks, 2016, 42 (C): 61-73.

[20] 黄庆东，闫乔乔，孙晴. 一种改进的 ad hoc 无线网络连通支配集生成方法 [J]. 电子科技大学学报, 2017, 46 (6): 819-824. (Huang Qingdong, Yan

Qiaoqiao, Sun Qing. An improved formation method of connected-dominating set in Ad hoc wireless networks [J]. Journal of University of Electronic Science and Technology of China, 2017, 46 (6): 819-824.)
[21] Touil S, Mahfoudhi, Laouamer L, *et al.* A novel scheme for selecting minimum connected dominating set in Ad hoc and WSNs [J]. Research Journal of Applied Sciences, 2017, 12: 409-415.
[22] Shukla S, Misra R, Agarwal A. Virtual coordinate system using dominating set for GPS-free adhoc networks [J]. Annals of Telecommunications, 2017, 72 (3-4): 1-10.
[23] Fomin F V, Grandoni F, Kratsch D. A measure & conquer approach for the analysis of exact algorithms [J]. Journal of the ACM, 2009, 56 (5): 1-32.
[24] 周晓清，肖鸣宇. 无向图中子集反馈顶点集问题的精确算法 [J]，计算机学报，2018, 41 (3): 493-505. (Zhou Xiaoqing, Xiao Mingyu. Exact Algorithms for the Subset Feedback Vertex Set Problem in Undirected Graphs [J]. Chinese Journal of Computers, 2018, 41 (3): 493-505.)
[25] Eppstein D. Quasiconvex analysis of multivariate recurrence equations for backtracking algorithms [J]. ACM Trans on Algorithms, 2006, 2 (4): 492-509.